**Project Title:**  
**AI-Based N-Puzzle Solver Using Search Algorithms**

**Submitted By:**  
Sameed Hasan (22k-4447)  
Muhammad Hamza Haroon (22k-4200)

**Course:**  
AI

**Instructor:**  
Muhammad Khalid Khan

**Submission Date:**  
[11-5-2025]

**1. Executive Summary**

**Project Overview:**  
This project presents an AI-based N-Puzzle solver implemented using multiple search algorithms such as A\*, Uniform Cost, Breadth-First Search (BFS), Depth-First Search (DFS), Iterative Deepening, and Greedy Best-First Search. The GUI-based solver supports puzzle dimensions ranging from 2x2 to 5x5, allowing users to input custom start and goal states. The primary objective is to compare the efficiency and correctness of various search algorithms in solving the N-Puzzle problem.

**2. Introduction**

**Background:**  
The N-Puzzle is a classic AI problem involving a grid of numbered tiles with one empty space. The goal is to move tiles around using the empty space to reach a desired goal configuration. This problem is a great candidate for testing heuristic search algorithms due to its defined state space and solvability conditions.

**Objectives of the Project:**

* Implement multiple search strategies to solve the N-Puzzle problem.
* Provide a graphical user interface for custom input and visualization.
* Compare algorithms based on nodes processed, memory usage, and time efficiency.
* Highlight the performance of heuristic-based approaches like A\*.

**3. Game Description**

**Original Game Rules:**

* The N-Puzzle consists of a grid with numbered tiles from 1 to N²−1 and one empty tile (represented as 0).
* A valid move consists of sliding an adjacent tile into the empty space.
* The objective is to reach the goal state from the initial state using a series of valid moves.

**Innovations and Modifications:**

* Customizable board size (2x2 to 5x5) through GUI.
* Visualization of start, goal, and solution states.
* Multiple search algorithm options for solving the puzzle.
* Step-by-step navigation through the solution path.

**4. AI Approach and Methodology**

**AI Techniques Used:**

* Breadth-First Search (BFS)
* Depth-First Search (DFS)
* Iterative Deepening
* Uniform Cost Search
* A\* Search with Manhattan Distance Heuristic
* Greedy Best-First Search

**Algorithm and Heuristic Design:**

* The Manhattan Distance heuristic was used in A\* and Greedy Best-First Search for estimating the cost to reach the goal.
* Each node keeps track of its state, depth (g), and parent for path reconstruction.
* A priority queue (heapq) is used for uniform cost, A\*, and greedy strategies.

**AI Performance Evaluation:**

* Metrics such as number of nodes processed, maximum nodes stored in memory, and time taken were displayed after each run.
* GUI allows visual tracking of moves to better understand the decision path.

**5. Game Mechanics and Rules**

**Modified Game Rules:**

* Any N x N configuration from 2x2 to 5x5 is allowed.
* The user must input a valid permutation of numbers from 0 to N²−1.
* Moves are only allowed between adjacent tiles and the empty space.

**Turn-based Mechanics:**

* The AI selects and applies moves in sequence based on the chosen strategy until the goal is reached.

**Winning Conditions:**

* The puzzle is solved when the current state exactly matches the goal configuration.

**6. Implementation and Development**

**Development Process:**

* Designed the Node and GoalTree classes to handle state expansion and search logic.
* Created a Tkinter-based GUI for user interaction and visualization.
* Integrated all search strategies and provided algorithm selection via dropdown.
* Tested with various puzzle sizes to validate correctness and performance.

**Programming Languages and Tools:**

* **Programming Language:** Python
* **Libraries:** NumPy, Tkinter, heapq, collections
* **Tools:** GitHub for version control

**Challenges Encountered:**

* Ensuring efficient state comparison and hashing for large puzzle sizes.
* Preventing memory overload in DFS and BFS for larger boards.
* Handling invalid user input robustly in the GUI.

**7. Team Contributions**

* **Muhammad Hamza Haroon (22k-4200):** Designed GUI, Developed and tested AI algorithms including A\*, BFS, DFS, Uniform cost and heuristic design.
* **Sameed Hasan (22k-4447):** Developed and tested AI algorithms including Iterative Deepening, Greedy Best-First Search.Designed GUI using Tkinter, managed user interaction, and integrated search logic with interface.

**8. Results and Discussion**

**AI Performance:**

* A\* with Manhattan Distance performed best in terms of efficiency and optimality.
* BFS guaranteed optimal solutions but used more memory.
* DFS was faster but risked getting stuck in loops for larger boards.
* For a 3x3 puzzle, A\* solved most cases within 1-2 seconds, processing fewer than 1000 nodes in typical scenarios.
* The results panel in the GUI provided clear feedback on each run, making performance evaluation user-friendly.

**9. References**

* Russell, S. J., & Norvig, P. (2021). *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson.
* NumPy Documentation
* Python Tkinter Documentation